DL4G - Questionnaire

Deep Learning for Games

Maurin D. Thalmann

20. Januar 2020

Dieser Questionnaire wurde basierend auf einer Card2Brain Sammlung erstellt:
Card2Brain - DL4G (Credits: Cyrille Ulmi)

Inhaltsverzeichnis

1 Sequenzielle Spiele

1.1 Was sind die Eigenschaften von endlichen-sequenziellen Spielen?

- Eine endliche Anzahl Spieler mit einer endlichen Anzahl Aktionen
- Die Aktionen werden sequenziell ausgewählt
- Es wird eine endliche Anzahl Runden gespielt
- Spätere Spieler sehen die Aktionen vorheriger Spieler

1.2 War wird unter Perfect Recall verstanden?

Perfekte Erinnerung an alle vorherigen Züge

1.3 Was ist eine Strategie?

Sagt einem Spieler, welche Aktion im aktuellen Zug auszuführen ist

1.4 Was ist ein Strategie-Profil?

Die ausgewählte Strategie eines Spielers

1.5 Was ist eine Utility- oder Payoff-Function?

Sie berechnet das Resultat für jede Aktion

1.6 Was sind die Komplexitätsfaktoren bei einer Spielanalyse?

- Anzahl Spieler
- Grösse des Suchraums (Anzahl gespielte Züge & Anzahl mögliche Aktionen)
- Kompetitiv vs. Kooperativ
- Stochastische Spiele (mit Zufall) vs. Deterministisch
- Perfekte vs. imperfekte Information

1.7 Was ist imperfekte Information?

- Das Spiel konnte nur teilweise beobachtet werden
- Man kennt bspw. nicht die Karten der anderen Spieler

1.8 Beispiele von Spielen mit perfekten / imperfekten Informationen?

Perfekt (Schach) und imperfekt (Jass, Poker)

1.9 Was ist der Suchraum?

Anzahl gültige Brettpositionen und die untere Grenze des Suchbaums

1.10 Was ist ein Suchbaum?

- Knoten sind Spielpositionen / Spielzustände
- Kanten sind Aktionen / Spielzüge
- Blätter werden durch Payoff-Funktionen definiert

1.11 Wie funktioniert Backward Induction?

- Den Baum von unten nach oben durcharbeiten (bzw. von rechts nach links)
- Immer den besten Weg für den aktuellen Spieler markieren
- Geeignet für sequenzielle endliche Spiele mit perfekter Information

1.12 Was bedeutet Rationalität?

Dass der Spieler nicht die schlechtere Alternative wählt

1.13 Welche Arten von Lösungen werden bei endlich-sequenziellen Spielen unterschieden?

- Ultra-schwache Lösung
  - Bestimmt, ob der erste Spieler einen Vorteil aus der Initialposition hat, ohne die genaue Strategie zu kennen
  - Setzt perfektes Spielen des Gegners voraus
  - Beispielsweise durch Existenzbeweise in der Mathematik
- Schwache Lösung
  - Kann ein komplettes Spiel mit perfekten Zügen aus der Initialposition durchspielen
  - Geht von einem perfekten Spiel des Gegners aus
- Starke Lösung
  - Kann aus jeder Position heraus perfekte Züge spielen
  - Kann auch gewinnen, wenn vorherige Spieler einen Fehler gemacht haben

1.14 Was versteht man unter einem Zero-Sum Game (Nullsummenspiel)?

- Der Vorteil für einen Spieler ist zum Nachteil des anderen Spielers
- Die Punktesumme für zwei Strategien ist immer gleich Null

1.15 Was sind Charakteristiken des Minimax-Algorithmus?

- Gilt nur für ein Nullsummenspiel
- Zwei Möglichkeiten / Ziele
  - den eigenen Gewinn maximieren
  - den Gewinn des Gegners minimieren

1.16 Wie funktioniert der Minimax-Algorithmus?

- Wenn der Knoten mir gehört: Aktion wählen, die den Payoff maximiert
- Wenn der Knoten dem Gegner gehört: Aktion wählen, die den Payoff minimiert
- Wenn es ein Endknoten ist: den Payoff berechnen

1.17 Was versteht man unter Search Tree Pruning?

Nicht relevante Teilbäume können weggelassen werden, reduziert den Rechenaufwand

1.18 Was sind die Regeln von Alpha-Beta Pruning?

- α ist der grösste Wert alles MAX Vorfahren eines MIN Knoten
- β ist der kleinste Wert alles MIN Vorfahren eines MAX Knoten
- Den Teilbaum abschneiden, falls er grösser als α oder kleiner als β ist

1.19 Was ist der Vorteil von Alpha-Beta Pruning?

- b = Anzahl Kanter der Knoten und m = Tiefe des Baums
- Ordnung verbessert sich von $O(b^m)$ nach $O(b^{m/2})$, halbiert also die Tiefe der Suchbäume

2 Monte Carlo Tree Search

2.1 Wieso werden Random Walks eingesetzt? (Tree Search)

- Der Suchraum ist oft zu gross für eine vollständige Suche
- Die Idee, verglichen zu Minimax, ist, bei einer bestimmten Tiefe zu stoppen und zu raten

2.2 Was ist die Idee hinter Monte Carlo Tree Search?

- Macht einen Random Walk und spielt zufällige Simulationen
- Versucht, in einer fixen Zeit möglichst viel des Suchraums zu entdecken
- Am Schluss wird der vielversprechendste Spielzug ausgewählt

2.3 Welche 4 Phasen gibt es bei Monte Carlo Tree Search?

1. **Selection**
   - Starte beim Wurzelknoten `R` und wähle fortlaufend Kinderknoten
   - Stoppe, wenn du einen Knoten erreichst, der noch nicht komplett erweitert/erforscht wurde
   - Benötigt ein Kriterum für die Auswahl der Kinderknoten, sogennante *tree policy*
2. **Expansion**
   - Wenn das Zeitlimit `L` das Spiel beendet, gib die Payoffs zurück
   - Sonst, wähle eine unerforschte Aktion und kreiere einen Knoten `C` für diese
3. **Simulation**
   - Simuliere ein Weiterspielen von Knoten `C` aus, mithilfe einer *default policy*
   - Im simpelsten Fall, spiele einfach bis zu irgendeinem Ende mit zufälligen Zügen
4. **Backpropagation**
   - Aktualisiere die gespeicherten Informationen in jedem Knoten von `C` zurück bis zu `R`
   - MCTS erwartet einen Payoff in [0,1]

2.4 Welche zwei Ansätze gibt es beim Auswählen eines neuen Knotens?

- Exploitation
  - Immer den besten Payoff wählen
  - Anhand von Beobachtungen auf der besten Maschine spielen, um Gewinn zu maximieren
- Exploration
  - Etwas Neues wählen, versuchen möglichst viel zu erkunden
  - Alle Maschinen spielen, um möglichst viel Informationen zu gewinnen

2.5 Was ist die Idee hinter UCB1 (Upper Confidence Bound)?

- Die beste Strategie ist eine Mischung aus Exploitation und Exploration
- Ergibt ein statistisches Konfidenzintervall für jede Option
- Parameter c kontrolliert den Trade-Off zwischen Exploitation und Exploration

2.6 Was ist sehr wichtig bei der Anwendung von UCB1?

Immer die Vektor-Komponente des aktuellen Spielers für die Berechnung verwenden.

2.7 Was passiert bei MCTS, wenn die Zeit abgelaufen ist?

Spielt die Aktion mit der höchsten Anzahl an Besuchen.

2.8 Was sind Unterschiede zwischen Minimax und MCTS?

- Beide Algorithmen setzen perfekte Informationen voraus
- Minimax ist nur anwendbar auf Nullsummenspiele mit zwei Spielern
- MCTS funktioniert für jedes Spiel mit perfekter Information
- Minimax optimiert Payoffs, MCTS optimiert einen Exploitation-Exploration Trade-Off
- MCTS ist ein Anytime-Algorithmus, Minimax nicht
- Monte Carlo Bäume sind asymmetrisch, Minimax Bäume sind symmetrisch

2.9 Was ist ein Anytime-Algorithmus?

Er kann eine gültige Lösung zurückgeben, auch wenn die Ausführung vorzeitig abgebrochen wird. Es wird erwartet, dass er eine immer bessere Lösung findet, je länger er ausgeführt wird.

2.10 Wie sehen die Payoffs bei MCTS aus und was wird maximiert?

- Für ein Beispiel mit 2 Spielern nimmt der Payoff-Vektor die Form $(W, N - W)$ an
- Spieler 1 maximiert W, Spieler 2 maximiert $N - W$ (implizit minimiert Spieler 2 so auch $-W$)

3 Information Sets

3.1 Was ist ein Information Set?

- Ein Information Set ist eine Menge von Knoten des gleichen Spielers
- Der Spieler kennt den vorherigen Zug nicht
- Alle Knoten müssen die gleichen Optionen bieten
- Sind immer aus der Sicht eines Spielers

3.2 Was ist der Unterschied zwischen perfekter und imperfekter Information?

- Unterschiedliche Strategien werden gewählt
- Bei perfekter Information hat jeder Knoten exakt eine Option

3.3 Was ist die Idee hinter Determinization?

- Unbekannte Karten zufällig auf die Gegner verteilen
- Danach das Anwenden der Regeln von perfekter Information (bspw. mit MCTS)
- Ergibt dann mehrere mögliche Suchbäume
- Schlussendlich über alle Bäume die Option wählen, die am meisten besucht wurde
- Manchmal werden Entscheidungen getroffen, welche gar nie eintreten können, was zu falschen Entscheidungen führen kann

3.4 Was muss bei einer Implementation von MCTS mit Information Sets beachtet werden?

- Der Baum besteht aus Information Sets und nicht mehr aus Zuständen
- Knoten entsprechen Information Sets aus der Sicht des Wurzelspielers
- Karten werden zufällig verteilt und ungültige Varianten werden ausgeblendet
- Kanten entsprechen einer Aktion, welche in mindestens einem Zustand möglich ist
- Funktioniert danach wie ein Spiel mit perfekter Information

3.5 Wie muss die UCB1 angepasst werden für Information Sets?

- N_p = Anzahl Besuche des Vorgänger-Knotens und wie oft Knoten i verfügbar war
- Knotenliste ergänzen um „wie viel mal war jede Option verfügbar“

4 Supervised Machine Learning

4.1 Welche Disziplinen gibt es bei Machine Learning?

- Supervised Learning
  - Dem Algorithmus werden gelabelte Trainingsdaten übergeben
  - Er lernt, Labels von unbenannten Daten vorherzusagen
- Unsupervised Learning
  - Dem Algorithmus werden nicht gelabelte Trainingsdaten übergeben
  - Er entdeckt / lernt selbstständig die Struktur der Daten
- Semi-Supervised Learning
  - Mischung zwischen Supervised und Unsupervised Learning
  - Wird meist benutzt, wenn eine kleine Anzahl gelabelter Daten zur Verfügung steht
- Reinforcement Learning
  - Keine Daten stehen zur Verfügung
  - Der Algorithmus wird durch eine Reward Funktion geleitet
  - Das ideale Verhalten wird gesucht, um die Reward Funktion zu maximieren

4.2 Was können die Gründe für schlechte Datenqualität sein?

- Schlechtes Design, mangelhafte oder inkonsistente Datenformate
- Programmierfehler oder technische Probleme
- Alter der Daten (bspw. ungültige E-Mail-Adressen)
- Schlechte Dateneingabemasken (fehlende Validierung bei Erfassung der Daten)
- Menschliche Fehler beim Datenexport
- Ungültige oder falsche Informationen

4.3 Welche Möglichkeiten gibt es, um die Qualität von Daten festzustellen?

- Datenquellen und deren Zuverlässigkeit überprüfen
- Statistische Kennzahlen interpretieren und überprüfen
- Visuelles Überprüfen eines Datenauszugs
- Manuell Datenbereiche überprüfen (bspw. negative Löhne)
- Plausibilität von Zusammenhängen überprüfen
- Redundanz der Daten messen
- Abweichungen in Syntax und Semantik der Daten
- NULL-Werte und doppelte Daten untersuchen

4.4 Wie läuft ganz einfaches Machine Learning ab?

- Daten in Testdaten und Trainingsdaten aufteilen
- Klassifikator auf den Trainingsdaten trainieren
- Klassifikator anhand der Testdaten bewerten

→ Funktioniert nur mit vielen Daten und wenn die Hyperparameter festgelegt sind

4.5 Wie sollten die Daten in einem Date Pool aufgeteilt werden?

- ca. 70-80% Trainingsdaten
- ca. 20-30% Testdaten

4.6 Wie sollte beim Aufteilen der Daten vorgegangen werden?

- Daten zufällig mischen
- Daten aufteilen in Training und Test
- Training- und Testdaten müssen disjunkt sein (keine gemeinsamen Elemente)

4.7 Was bedeutet Training? (ML)

Minimierung einer Kostenfunktion auf den Trainingsdaten durch Anpassen der Modellparameter

4.8 Was bedeutet Testing? (ML)

Nur eine Auswertung auf unbekannten Daten ermöglicht, die Leistung eines Modells festzustellen

4.9 Was ist k-NN?

- Steht für K-Nearest Neighbors
- Ist ein sehr einfacher Machine-Learning-Algorithmus
- Bei $k = 1$ wird das Label des nächstgelegenen Trainingspunkt verwendet
- Bei $k > 1$ findet ein Mehrheitsbeschluss der nächsten k Nachbarn statt

4.10 Was sind Hyperparameter?

- Entscheidungen, die vom Menschen getroffen werden
- Zum Beispiel: Angleichung an eine lineare Funktion

4.11 Was sind Beispiele für Hyperparameter?

- Anzahl Nachbarn bei k-NN
- Regularisierung der Parameter
- Kernel einer Support-Vektoren-Maschine
- Baumtiefe und Selection-Policy im Entscheidungsbaum
- Anzahl Layers, Neuronen, Aktivierungsfunktion, Dropout bei Deep Learning

4.12 Wie sieht ein komplizierter Auswertungsvorgang aus? (ML)

- Daten aufteilen in 60% Training, 20% Validation und 20% Test
- Über alle interessanten Hyperparameter-Konfigurationen iterieren
- Modell anhand der ausgewählten Hyperparameter auf den Trainingsdaten trainieren
- Modell auf dem Validation-Set überprüfen und Leistung messen
- Modell mit der besten Leistung auswählen und auf den Testdaten Leistug messen

$\rightarrow$ Diese variante benötigt ziemlich viele Daten

4.13 Was ist das Ziel von Cross Validation?

- Falls zu wenig Daten vorhanden sind
- Aufteilen in 80% Training und 20% Test
- Training aufteilen in k Folds (bspw. 10)
- Jeder Fold ist einmal das Validation-Set
- Die restlichen Folds werden als Trainingsdaten verwendet
- Schlussendlich den Mittelwert über alle Folds ziehen

4.14 Was ist das Ziel bei Aufteilung in Test- und Trainings-Set?

Gleiche Verteilungen über Test- und Trainings-Set

4.15 Was sollte man über Learning-Curves wissen?

- Wenn Kurven weit auseinander sind, hilft es wenn man mehr Daten nutzt
- Wenn Kurven parallel sind, dann bringt es ziemlich sicher nichts

Vorhersage	Wahr	Falsch
Realität: *Wahr*	TP	FN
Realität: *Falsch*	FP	TN

4.16 Welche Möglichkeiten gibt es bei einer Confusion Matrix für einen Binary Classifier?

TN = true-negative
FP = false-positive
FN = false-negative
TP = true-positive

4.17 Wie werden Accuracy und Error Rate berechnet?

- Accuracy = $\frac{TP+TN}{Total}$
- Error Rate = $\frac{FP+FN}{Total} = 1 - Accuracy$

5 Neuronal Networks

5.1 Was ist der Unterschied zwischen AI und Deep Learning?

- Artificial Intelligence (regelbasiert / MCTS)
- Machine Learning (Training auf Daten mit festgelegten Features)
- Deep Learning (Lernen der Features aus den Daten)

5.2 Was ist die Definition von Machine Learning?

- Ein Computerprogramm **lernt** aus der Experience E...
- ...mit Rücksicht auf eine Klasse von Tasks T und Performance Measures P...
- ...wenn es Tasks in T durchführt, gemessen an P, seine Experience E verbessert.

5.3 Was ist T beim Jassen? (ML)

Tasks (T)
- Trumpf auswählen
- Karte auswählen
- Werte der Karten
- Punkte im Spiel

5.4 Was ist P beim Jassen? (ML)

Performance Measure (P)
- Anzahl Gewinne / Punkte
- Vergleich mit Daten

5.5 Was wäre E beim Jassen? (ML)

Experience (E)
- Spielen
- Daten lesen

5.6 Welche zwei Task-Typen werden unterschieden? (ML)

Regression & Classification

5.7 Aus welchen 3 Teilen besteht ein Feed Forward Network?

- Input Layer
- Hidden Layer
- Output Layer

5.8 Was beinhaltet ein Knoten eines Netzwerks?

- Inputs (x_1, x_2, ...)
- Internen Parameter (Θ)
- Funktion, die den Output basierend auf dem Parameter berechnet

5.9 Was ist eine Epoche?

Ein (1) Training des gesamten Datensets

5.10 Welche Aktivierungsfunktionen gibt es?

- Sigmoid für eine binäre Klassifizierung
- relu für Nodes innerhalb eines Netzwerks
- Aktivierungsfunktion sollte nicht linear sein

5.11 Was ist eine Kostenfunktion?

- Ist die Funktion, welche durch das Lernen minimiert werden soll
- Häufig wird eine Likelihood-Funktion verwendet

5.12 Was sollte bei Multi-Class Problemen beachtet werden?

- 1-hot encoded Array für Labels
- Letzter Layer mit softmax Funktion (für Normierung)

5.13 Was ist ein neuronales Netzwerk?

- Besteht aus mathematischen Knoten
- Knoten haben Inputs und Outputs
- Knoten sind organisiert in Layers
- Knoten berechnen aus Inputs und Parameter einen Output
- Knoten haben lineare Funktion (Gewicht, Bias, Input)
- Auf Funktion wird eine Aktivierungsfunktion angewendet (sigmoid, relu)

5.14 Wie berechnet ein neuronales Netzwerk das Resultat?

- Parameter werden während dem Training definiert
- Danach nur noch Layer für Layer durchrechnen

5.15 Was ist eine Loss Funktion?

- Grundsätzlich eine Fehlerfunktion
- Die Funktion, welche während dem Training minimiert werden soll

5.16 Wie wird ein neuronales Netzwerk trainiert?

- Differenz zwischen berechnetem Wert und gegebenem Label
- Es wird versucht, Differenzen zu minimieren

5.17 Wie kann das Training auf seine Wirksamkeit überprüft werden?

- Benötigt eine Metrik und Testdaten
- Zum Beispiel: Überprüfen der Accuracy

5.18 Was ist der Vorteil eines Deep Neuronal Networks?

Features werden automatisch berechnet

5.19 Was ist das Problem mit der optimalen Kapatität? (NN)

- Mit mehr Trainingsdaten wird der Fehler kleiner
- Der Fehler der Testdaten wird dann jedoch immer grösser

5.20 Welche 3 Klassen gibt es bei VOC Challenges?

- **Klassifizierung**: Enthält das Bild ein Objekt einer Klasse?
- **Erkennung**: Welche Klasse haben die Objekte eines BIldes?
- **Segmentierung**: zu welcher Klasse gehört ein einzelnes Pixel?

5.21 Was ist die Funktion einer Support Vector Machine?

- Berechnet Features und Deskriptoren
- Berechnet eine Art Fingerabdruck auf dem Bild

5.22 Was ist die Idee hinter einem Residual Network?

- Layer darf das Ergebnis nicht verschlechtern
- Falls Layer nichts bewirkt, einfach ignorieren

5.23 Welche Layer-Typen gibt es? (NN)

- Dense / Fully-Connected
  - Jeder Knoten ist mit jedem Knotem aus dem Layer davor verbunden
  - Jede Verbindung hat ein Gewicht
- Convolutional
  - Knoten ist nur mit einem Teil der vorherigen Knoten verbunden
  - Immer die gleichen Gewichtungen verwenden
- Polling
  - Berechnet das Maximum oder den Durchschnitt einer Region
  - Macht das Bild kleiner

5.24 Was ist Regularisierung?

- Zur Loss-Funktion wird eine Penalty-Funktion addiert
- Penalty-Funktion ist abhängig von den Gewichten

5.25 Was ist L2-Regularisierung?

Quadrat der Gewichte zur Loss-Funktion addieren

5.26 Welche anderen Regularisierungs-Methoden gibt es?

- Ensemble Method
  - Abstimmung von mehreren separat trainierten Netzwerken
- Dropout
  - Zufällige Knoten mit 0 multiplizieren
  - Netzwerk muss mehrere Wege finden
- Early Stopping
  - Stoppen, wenn Fehler des Validation-Sets zu wachsen beginnt
  - Lernen der benötigten Anzahl Trainingsschritte (Hyperparameter)

5.27 Was ist der Unterschied zwischen einem Non-Deep und Deep Neural Network?

- Deep Neural Network erkennt die Features selbstständig
- Es werden nur noch die Daten benötigt

5.28 Was ist ein Convolutional Neural Network?

- Es sind nicht alle Knoten mit dem vorherigen Layer verbunden
- Die Gewichte werden für jede Position geteilt

5.29 Was bedeutet Regularisierung und wieso wird es gebraucht?

- Penalty auf den Gewichten
- Zum Beispiel: Quadrate der Gewichte zur Loss-Funktion addieren
- Um eine glättere Kurve zu bekommen
- Verhindert Overfitting

5.30 Was bedeutet Dropout und wieso wird es gebraucht? (RL)

- Verwirft zufällig Knoten (setzt sie auf 0)
- Zwingt das Netzwerk, Alternativen zu finden
- Gibt ein stabileres Netzwerk

5.31 Was ist die Idee hinter Inception und Res-Net?

- Inception
  - Mehrere Layer nebeneinander und dann addieren
- Res-Net
  - Shortcut zwischen einzelnen Knoten
  - Einfacheres Lernen von komplizierten Netzwerken

5.32 Was ist Reinforcement Learning?

- Ziel: Reward maximieren
- Reward kann durch Aktionen verändert werden
- Aktionen beeinflussen die Umgebung

5.33 Was ist Reinforcement Learning nicht?

- Supervised Learning: es gibt keinen Supervisor, der sagt, was richtig ist
- Unsupervised Learning: es werden auch keine Strukturen gesucht

5.34 Was sind die Eigenschaften von Reinforcement Learning?

- Verzögertes Feedback
- Zeit ist relevant
- Aktionen beeinflussen die Umgebung und die Daten

5.35 Was ist das Ziel von Reinforcement Learning?

Ein Reward Rt ist ei Feedback, welches angibt, wie gut sich der Agent zum Zeitpunkt t verhält.

5.36 Welche Arten von RL-Agenten gibt es?

- Value Based: no policy (implicit), value function
- Policy Based: policy, no value function
- Actor Critic: policy, value function
- Model Free: policy and/or value function, no model
- Model: policy and/or value function, model

5.37 Welche Varianten von Greedy Algorithmen gibt es?

- greedy
  - Aktion wählen mit dem maximal geschätzten Wert
- non greedy
  - Zufällige Aktion wählen
- e-greedy
  - Mit Wahrscheinlichkeit beliebige Aktion auswählen
  - Mit 1- Wahrscheinlichkeit maximale Aktion auswählen

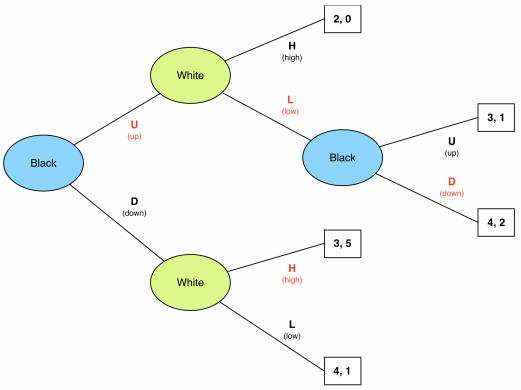
5.38 Wie kann eine Policy ausgearbeitet werden?

- Beginnen mit einer zufälligen Policy
- Berechnen der Value-Funktion v unter Berücksichtigung der aktuellen Policy
- Policy verbessern durch greedy-Auswahl der Aktionen aus v
- Wiederholen, bis sich die Policy nicht mehr verändert

6 Weitere Fragen

6.1 Wie funktioniert Backward Induction?

- Den Baum von unten nach oben durcharbeiten (oder eben von rechts nach links)
- Immer den besten Weg (mit dem höchsten Payoff) für den aktuellen Spieler markieren
- Ist für sequenzielle endliche Spiele mit perfekter Information geeignet



6.2 Wie funktioniert der Minimax-Algorithmus?

- Wenn der Knoten mir gehört: Aktion wählen, die den Payoff maximiert
- Wenn der Knoten dem Gegner gehört: Aktion wählen, die den Payoff minimiert
- Wenn es ein Endknoten ist: Den Payoff berechnen

6.3 Was ist die Formel für UCB1?

$$U_i = \frac{W_i}{N_i} + c\sqrt{\frac{\ln N_p}{N_i}}$$

- W_i = Anzahl Gewinne mit der Maschine i
- N_i = Anzahl Versuche mit der Maschine i
- N_p = Anzahl Versuche insgesamt

6.4 Erklären Sie den Algorithmus, der 1997 im Schach gewonnen hat.

Deep Blue

- 8000 handgefertigte Features
- Evaluation des Bretts mit dem Skalarprodukt zwischen Features und Gewichten
- Gewichte wurden vor allem von Hand angepasst durch menschliche Experten
- High-Performance parallele Minimax mit Alpha-Beta Pruning
- 480 spezialgefertige VLSI Schach-Prozessoren
- Durchsucht 200 Millionen Positionen pro Sekunde

6.5 Wie heissen die 4 Phasen bei MCTS und wie würden diese aussehen?